CMPSC265 Data Structures and Algorithms
Homework 2

Due: Sunday, Sept 15th, 2019, 11:59pm

Learning Objectives

- Arrays and ordered arrays
- Binary search
- Time complexity analysis and big-O notation

Deliverables:

- A .doc or .pdf containing your answers to Problem 1.
- The modified java file *HighArray.java* for Problem 2
- The modified java file *OrdArray.java* for Problem 3
- A java file *AddingTwo.java* for Problem 4

Instructions:

- For Problem 2 and 3, please first download the Java source file, modify the codes based upon requirements, and submit the updated source file.
- Please add the required *credit comments* and *comments* to each Java program you submit.

Problem Specifications:

Problem 1: Time analysis and Big-O Notation (20')

For the following two Java code segments, please analyze its time complexity and represent it in terms of Big-O notation.

1.1

```
int sum = 0;
for (int i = 1; i < N; i *= 2)
   for(int j = 0; j < i; j++)
      sum++;
```

1.2

```
int sum = 0;
for (int i = 1; i < N; i *= 2)
   for (int j = 0; j < N; j++)
      sum++;
```

Problem 2: ***HighArray class (25')***

Description:

Please first download the attached HighArray.java file. You are expected to implement the *removeMax()* method, so that the item in the array with the largest value can be returned, and it will also be removed from the array. Return -1 if the array is empty. If there are duplicate maximum values, please remove all of them.

Outputs:

Your outputs should look something as follows.

```
The original array is:
77 99 44 55 22 88 11 0 99 66 33
The maximum value is 99
The array after removing the maximum value is:
77 44 55 22 88 11 0 66 33
The maximum value is 88
The array after removing the maximum value is:
77 44 55 22 11 0 66 33
The maximum value is 77
The array after removing the max value is:
44 55 22 11 0 66 33
```

Problem 3: OrdArray class (25')

Description:

Please first download the attached OrdArray.java file. You are expected to:

- Implement the *merge()* method to the OrdArray class so that you can merge two ordered arrays into an ordered destination array and return that array.

- Write code in the *main()* method to create two ordered arrays with arbitrary values, invoke the *merge()* method, and display the contents of the resulting destination array.

Outputs:
Your outputs should look something as follows.

```
The first ordered array is:
1 3 5 5 7 9 11
The second ordered array is:
2 4 6 8 10 10
The merged ordered array is:
1 2 3 4 5 5 6 7 8 9 10 10 11
```

Problem 4: AddingTwo class (30')

Description
Please write a Java program to first read in an array of integers with elements obtained from keyboard. Make sure all the inputs are in ascending order, so that the created array is an ordered array. Given any target number, find in the array two elements that can add up to the target number. Return and display the *index* of the two elements onto screen. Return {-1, -1} if no such two elements can be found in the array.

***Note**: you can get the full 30' credit for this problem is your implementation is in O(N), and 20' credit if your implementation is in $O(N^2)$, where N is the length of the array.*

***Hint:** since the input array is ordered, try to apply an idea similar to binary search (with two moving pointers).*

Specification
Two methods are needed in this AddingTwo class.

- main() method
- addingTwo() method.

main() method:

1) Please read in an array of integers of arbitrary lengths and elements from keyboard. Make sure the inputs are in ascending order.

2) Ask and get the input of an arbitrary target number.
3) Invoke the addingTwo() method, and get the returned indexes.
4) Display the index on screen.

addingTwo() method

- The method has two parameters: an array of integers, and an int for the target number.
- The method will return an array of two numbers, indicating the two indexes of the numbers that can add up to the target number.
- Return {-1, -1} if no such two elements exist.
- If there is more than one solution, return any one should be good.

Outputs: Your results should look similar as follows.

$ java AddingTwo

```
How many elements in the array?: 7
Please enter one element's value: 6
Please enter one element's value: 4
Please enter one element's value: 10
Please enter one element's value: 2
Please enter one element's value: 5
Please enter one element's value: 1
Please enter one element's value: 0
Please enter a target value: 12
The two elements that add up to the target value are at
index: 2 3
```

$java AddingTwo

```
How many elements in the array?: 7
Please enter one element's value: 6
Please enter one element's value: 4
Please enter one element's value: 10
Please enter one element's value: 2
Please enter one element's value: 5
Please enter one element's value: 1
Please enter one element's value: 0
Please enter a target value: 20
The two elements that add up to the target value are at
index: -1 -1
```